Substituting ReLUs with Hermite Polynomials gives faster convergence for SSL

Anonymous ICCV submission

Paper ID 6415

Abstract

Rectified Linear Units (ReLUs) are among the most widely used activation function in a broad variety of tasks in vision. Recent theoretical results suggest that despite their excellent practical performance, in various cases, a substitution with basis expansions (e.g., polynomials) can yield significant benefits from both the optimization and generalization perspective. Unfortunately, the existing results remain limited to networks with a couple of layers, and the practical viability of these results is not yet known. Motivated by some of these results, we explore the use of Hermite polynomial expansions as a substitute for ReLUs in deep networks. While our experiments with supervised learning do not provide a clear verdict, we find that this strategy offers considerable benefits in semi-supervised learning (SSL) settings. We carefully develop this idea and show how the use of Hermite polynomials based activations can yield improvements in pseudo-label accuracies and sizable financial savings (due to concurrent runtime benefits). Further, we show via theoretical analysis, that the networks (with Hermite activations) offer robustness to noise and other attractive mathematical properties.

1. Introduction

Rectified Linear Units (ReLUs) are often the defacto activation function of choice in many modern deep learning architectures in computer vision. Compared to a number of other activations such as sigmoid, ReLUs are easier to compute, which often leads to faster training time. Experimentally, since ReLUs are linear in the positive quadrant, they do not saturate which helps mitigate the vanishing gradient problem common in other activation functions. Finally, since ReLUs are zero in the negative quadrant, one finds that many nodes in the network may not activate – this leads to sparse set of activations that is a desirable behavior [17, 27, 8] in general including accelerate the convergence of SGD [23] and regularization type benefits [13].

Some known problems with ReLUs. Despite the attractive properties of ReLUs including how well it behaves [23] with stochastic gradient descent and its variants like RMSProp or Adam, commonly used optimization techniques in the community, on the flip side, a number of practitioners find that ReLUs also raise certain problems, whose severity can vary based on the task at hand. For example, there is consensus that ReLUs suffer from the "dying ReLU" problem. Even ignoring pathological cases, it is not uncommon to find that as much as 40% of the activations could "die". This happens because of the functional form of ReLU which is zero for an input when it is non-positive. A direct consequence of this behavior is that the dead neurons will stop responding to variations in error/input. Further, ReLU also tends to "blow up" the activation when the output of the activation is not constrained. This makes ReLUs not very suitable for Recurrent Network based architectures such as LSTMs [24].

Potential solutions in the literature. The foregoing issues with ReLUs have been known for some time in the community and as a result, a number of papers have explored nice strategies to address these shortcomings – various proposals are now available. In general, the solutions seek to *relax* the functional form of a ReLU function to allow negative values to pass through the unit. For example, a Leaky ReLU modifies the function to allow small negative values when the input is less than zero [17]. The Exponential Linear Unit, or ELU, is a generalization of ReLU that uses a parameterized exponential function to transition from the positive to small negative values [7]. The Parametric ReLU, or PReLU, proposes learning parameters that control the shape of the function. Another option, Randomized ReLU (or RReLU) proposed in [43] is to calculate the negative slope randomly in each training iteration, based on the idea that a random negative slope mitigates overfitting. Concatenated ReLU (or CReLU) instead suggests using two outputs comprising of a ReLU and a negative form of ReLU. So, for a when an `input` is positive, CReLU produces $[\texttt{input}, 0]$, and when an `input` is negative, CReLU produces $[0, \texttt{input}]$. The two outputs lead to an increase in the output dimension. Leaky ReLU, PReLU and RReLU do not always provide noise-robust deactivation: this is because the negative component includes a slope; in contrast,

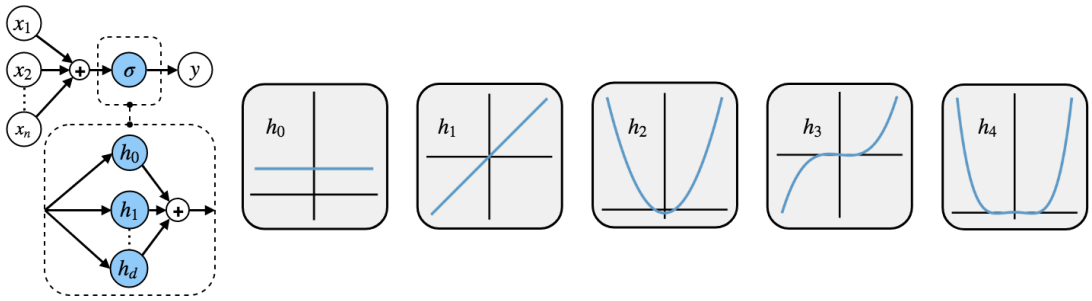


Figure 1: Hermite Polynomials as Activations (**leftmost**): Incorporating Hermite Polynomials as an activation function in a single hidden unit one hidden layer network. (middle) The functional form of first 5 hermites are shown in the **right**.

the original ReLU or ELU saturates in the negative range. Maxout is an alternative piecewise linear function that returns the maximum of the inputs, designed to be used in conjunction with the dropout regularization technique [14]. Some libraries also provide ReLU variants such as ReLU-6 where the output of ReLU is thresholded at an upper bound (e.g., 6) although other values can also be specified.

Recent results on the theoretical side. Until a few years back, the literature did not provide a great deal of guidance on the learning theoretic or optimization-centered properties of ReLUs other than the rather straightforward observation that the non-smoothness could be problematic from the optimization perspective. In [16], the authors demonstrated that residual networks with ReLU activation functions in particular have universal finite-sample expressivity. In other words, the model can represent any function provided that the model has more parameters than the sample size. That paper also derived a simple architecture consisting of only residual convolutional layers and ReLU activations, but no batch normalization, dropout, or max pooling. In other words, the choice of activations enabled simplification. Critical point and convergence analysis for two-layered ReLU networks has also been derived [41]. More recently, results in [42] give interesting insights into how over-parameterization using multi-layer feedforward ReLU based networks can help generalization. Specifically, the global minimizer of a weakly regularized cross-entropy loss also has the maximum normalized margin for two layer networks. Within the last year, [11] investigated optimizing the population risk of the loss using stochastic gradient descent and showed for one hidden layer network, one could avoid spurious local minima by utilizing an orthogonal basis expansion for ReLUs. A key takeaway from this work is that the optimization would behave better if the landscape was nicely behaved — this is enabled via the basis expansion. The foregoing results and analyses, especially the use of **basis expansion**, is interesting and indeed a starting point for the development described here. But a common feature of the body of work summarized above is that the results are limited mostly to networks with a small (or in some cases, one or two) number of layers.

Gap in literature: Relatively less is known whether this strategy or its variants are a good idea for the architectures in broad use in computer vision today. We can, in fact, ask a more focused question: are there specific tasks in vision where such a strategy offers strong practical advantages? This is precisely the gap this paper is designed to address.

The **main contributions** of this paper include: **(a)** We describe mechanisms via which activation functions based on Hermite polynomials can be utilized within deep networks instead of ReLUs, with only minor changes to the architecture. **(b)** We present evidence showing that while these adjustments are not significantly advantageous in supervised learning, our scheme *does* yield sizable advantages in semi-supervised learning speeding up convergence. Therefore, it offers clear benefits in compute time (and $) needed to attain a certain pseudo-label accuracy, which has direct cost implications. **(c)** We give technical results analyzing the mathematical behavior of such activations.

2. Brief Review of Hermite polynomials

We will use an expansion based on Hermite polynomials as a substitute for ReLU activations. To describe the construction clearly, we briefly review the basic properties.

Hermite polynomials are a class of orthogonal polynomials which are appealing both theoretically as well as in practice, e.g., radio communications [4, 28]. Here, we will use Hermite polynomials [35] defined as,

$$H_i(x) = (-1)^i e^{x^2} \frac{d^i}{d^i x} e^{-x^2}, i > 0; H_0(x) = 1 \quad (1)$$

In particular, we use normalized Hermite polynomials which are given as $h_i = \frac{H_i}{\sqrt{i!}}$. Hermite polynomials are often used in analysis of algorithms for nonconvex optimization problems [29, 22]. While there are various mathematical properties associated with Hermites, we will now discuss the most important property for our purposes.

Hermite polynomials as bases. Classical results in functional analysis show that the (countably infinite) set $\{H_i\}_{i=0}^{d}$ defined in (1) can be used as bases to represent smooth functions [37]. Formally, let $L^2(\mathbb{R}, e^{-x^2/2})$ denote the set of integrable functions w.r.t. the Gaussian measure

The normalized Hermite polynomials form an *orthonormal basis* in $L^2(\mathbb{R}, e^{-x^2/2})$ in the sense that $\langle h_i, h_j \rangle = \delta_{ij}$. Here δ_{ij} is the Kronecker delta function and h_i, h_j are any two normalized Hermite polynomials.

Recently, as described in Section 1, the authors in [12] showed that the lower order terms in the Hermite polynomial series expansion of ReLU has a different effect on the optimization landscape than the higher order terms for one hidden layer networks. Here, we investigate whether these properties are useful to accelerate the performance of gradient based methods for deep networks as well.

Hermite Polynomials as Activations. Let $x = (x_1, \ldots, x_n)$ be an input to a neuron in a neural network and y be the output. Let $w = (w_1, w_2, \ldots, w_n)$ be the weights associated with the neuron. Let σ be the non-linear activation applied to $w^T x$. Often we set $\sigma = ReLU$. Here, we investigate the scenario where $\sigma(x) = \sum_{i=0}^{d} c_i h_i(x)$, denoted as $\sigma_{hermite}$, were $h_i's$ are as defined previously and $c_i's$ are **trainable parameters**. As theory suggests, we initialized the parameters $c_i's$ associated with hermites to be $c_i = \hat{\sigma}_i$, where $\hat{\sigma}_i = \langle ReLU, h_i \rangle$. Hermite polynomials as activations on a single hidden unit single layer neural network can be visualized in Figure 1

3. Sanity check in the supervised setting: What do we gain and what do we lose?

Replacing ReLUs with other activation functions reviewed in Section 1 has been variously attempted in the literature already, for supervised learning. While improvements have been reported in specific settings, ReLUs continue to be relatively competitive. Therefore, it seems that we should not see expect improvements in what are toy supervised learning experiments. However, as we describe in more detail below, the experiments yield very useful feedback that provides an important hint regarding settings where Hermites will be particularly useful.

A) Two layer architectures. We start with the CIFAR10 dataset and a simple two-layer network (more details are given in the supplement). Our goal was to ask if this modification will lead to better or worse performance in accuracy and runtime (over ReLU activation) with all other parameters (e.g., learning rates) fixed.

Positive results: Here, we observed *faster* training loss convergence (over the initial epochs) compared to ReLU in general. *Negative results:* When we assessed the performance as a function of the number of Hermite polynomials d, we see a tradeoff where the speeds first improve with increasing d and then worsen when d is as high as 8, indicating that too many bases, especially for relatively shallow architectures is not ideal.

B) Key adjustments for deeper architectures. When implemented naively, Hermite activations do **not** work well for deeper architectures directly, which may be a reason that they have not been carefully explored so far. Basically, if no adjustments are used, we encounter a number of numerical issues that are not easy to fix. In fact, [13] explicitly notes that higher order polynomials tend to make the activations unbounded making the training unstable. Fortunately, a simple trick mentioned in [3] in the context of quadratic functions, addresses the problem. The solution is to add a softsign function in (3), which has a form similar to tanh however, it approaches its maximal (and minimal) value more slowly than tanh.

$$\text{Softsign}(x) = \frac{x}{1 + |x|} \tag{3}$$

C) ResNet18. With the above modification in hand, we can use the activation function within Resnet18 using Preactivation Blocks [18, 19]. In the preactivation block, we observed empirically that having the second softsign function after the weight layer is useful. The slightly modified preactivation block of ResNets is shown in Figure 2. We train

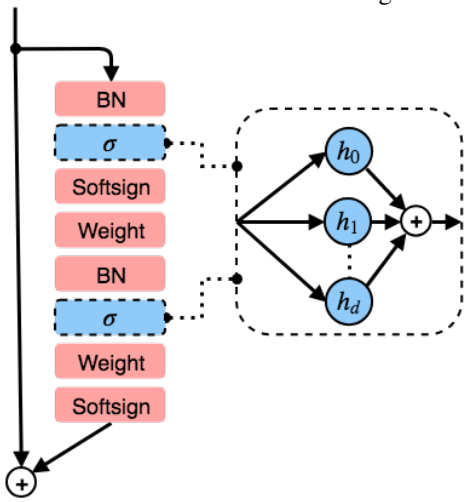
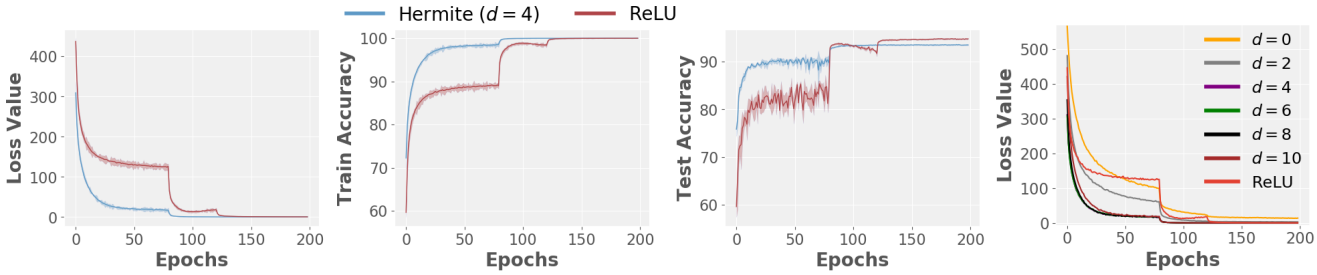


Figure 2: Hermite Polynomials as Activations in ResNets. We introduce softsign function into the residual block to handle the numerical issues arising due to the unbounded nature of the hermite polynomials.

ResNets with Hermite activations on the CIFAR10 dataset to assess the general behavior of our substitution. We use SGD as an optimizer and follow the data augmentation techniques and the learning rate schedules as per recommendations in [18, 5]. We perform cross-validation for the hyperparameters. Upon training, we obtain the loss curve and the training set accuracies for Hermite activations and ReLUs as shown in Fig. 3.

Positive results: We observe from the Figure 3 that the loss values and trainset accuracies converge at a much faster rate for Hermites than ReLUs. While the testset accuracy at the *final* epoch is higher for ReLU than Hermite activations, the



(a) Loss and Accuracy curves

(b) Effects of changing d

Figure 3: Hermites vs. ReLUs on ResNet18. (a) Hermites provide faster convergence of train loss and train accuracies than ReLUs. Hermites have faster convergence in test accuracies over the initial epochs but ReLU has the higher test accuracy at the end of training. (b) As we increase the number of hermite polynomials, the speed of loss convergence increases until $d = 6$ and then it starts to reduce. $d \geq 1$ performs better than $d = 0$ where only softsign is used as an activation.

testset accuracies for networks using Hermite activations make much quicker progress in the initial epochs. These results hold even when varying the learning rates of ReLU networks (see supplement).

Negative results We also tune the choice of the number of Hermite polynomials d and experiment with d in $0, 2, 4, 6, 8, 10$. The setting $d = 0$ is the case where we only use a softsign as an activation without any Hermite activations. Figure 3b shows the results of this experiment. From the plot, we observe a trend similar to the two layer network above, where the convergence speeds first improves and then reduces as we increase the number of Hermite polynomials. The setting $d = 0$ performs worse than when d is at least one, suggesting that the performance benefits is due to Hermite activations with softsign (and not the softsign function on its own).

D) Interesting take away from experiments? Let us consider the negative results first where we find that a large number of bases is not useful. This makes sense where some flexibility in terms of trainable parameters is useful, but too much flexibility is harmful. Therefore, we simply need to set d to a small (but not too small) constant. On the other hand, the positive results from our simple experiments above suggest that networks with Hermite activations make rapid progress in the early to mid epoch stages – an **early riser** property – and the performance gap becomes small later on. This provides two potential strategies. If desired, we could design a hybrid optimization scheme that exploits this behavior. One difficulty is that initializing a ReLU based network with weights learned for a network with Hermite activations (and retraining) may partly offset the benefits from the quicker progress made in the early epochs. What will be more compelling is to utilize the Hermite activations end to end, but identify scenarios where this "early riser" property is critical and directly influences the final goals or outcomes of the task. It turns out that recent developments in semi-supervised learning satisfy exactly these conditions where leveraging the early riser property is immensely beneficial.

4. Semi-Supervised Learning (SSL)

A general SSL procedure. To set up the algorithmic parts of our work, it will be helpful to review a semi-supervised learning (SSL) scenario and one potential general-purpose solution strategy, which can be described completely independent of Hermite activations. Let us assume that we are interested in image classification, and that we have access to large amounts of unlabeled images. In this setting, it makes sense that we may want to use the unlabeled examples to improve the performance of our classifier. One way to approach this problem would be to run a **two phase procedure**. In *Phase I*, we may estimate *pseudolabels* for the unlabeled images. Then, in *Phase II*, using the estimated pseudolabels, we may train any standard classifier such as ResNets on the entire dataset, in a completely supervised manner [25, 33]. The reason a discussion of this generic scheme helps in this context is simply to appreciate that the success of the two step procedure depends on both the quality of the pseudolabels and the ease of training a model (say, a ResNet) in the second step. In the following sections, we make the general procedure above a little more specific, and describe a framework where Hermites can provide significant computational/resource benefits.

The SaaS framework. It is well known that poor (random) labels are harder to fit (train) [45], or equivalently, high quality (pseudo) labels accelerates the training procedure. In other words, highly accurate labels $\rightarrow$ fast training. Interestingly, [6] showed that the converse statement, i.e., fast training $\rightarrow$ accuracy of labels, is *also* true using empirical evidence, during training. Using this result, the authors proposed a framework called SaaS (**Speed as a Supervi-**

sor) to *estimate* the pseudolabels using "speed" as a surrogate. That is, for a classifier such as ResNet, SaaS takes advantage of the fact that the *loss decreases at a faster rate for correct labels* (compared to random labels) during training. Furthermore, the *rate of loss reduction decreases as the percentage of incorrect labels in the dataset increases*.

In the primal phase (inner loop), SaaS seeks to find a set of pseudolabels that decreases the loss function over a *small number of epochs*, the most. In the dual phase (outer loop), the "pseudolabel" optimization is carried out by computing a posterior distribution over the unlabeled data. Specifically, the algorithm consists of two loops: (i) in the outer loop, we optimize over the posterior distribution of the pseudolabels, and (ii) in the inner loop, we retrain the network with fixed pseudolabels. After every inner loop, we reinitialize the network, and compute the rate of change of the loss value with which the posterior can be computed. A flowchart of the algorithm is shown in Figure 4. We can easily use a ResNet/DenseNet model in the primal phase. There are two different loss functions that are optimized during traning: the cross entropy loss (L_{CE}) over the labeled data, the unlabeled data and an entropy regularizer (Reg_E) on the pseudo-labels. A pseudocode is provided in Algorithm 1.

Pseudolabels with Hermites. Recall that networks with Hermite activations manifest the "early riser" property. Thsis turns out to be ideal in the setting described above and next, we show how this property can be exploited for semi-supervised learning. Intuitively, the early riser property implies that the training loss decreases at a **much faster rate** in the initial epochs. This is expected, since the optimization landscape, when we use Hermites are, by definition, *smoother* than ReLUs, since *all* the neurons are *always* active during training with probability 1.

Setting up. For our experimental purposes, we used a ResNet-18 architecture (with a preactivation block) [19]

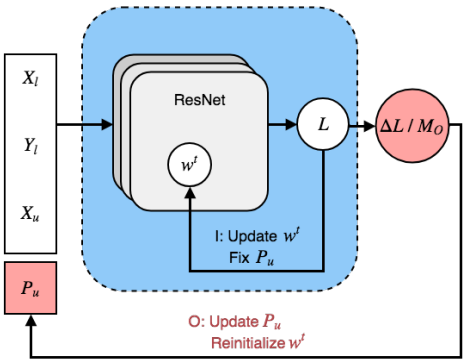


Figure 4: Illustration of the SaaS Framework. SaaS runs over two loops, in the inner loop denoted by I and the outer loop denoted by O.

Algorithm 1

Input: Labeled data (x_i, y_i), unlabeled data (z_i), number of classes k, #-outer (inner) epochs $M_O(M_I)$, loss function $L = L_{CE}(x_i, y_i) + L_{CE}(z_i, y_i) + Reg_E(z_i, y_i)$, learning rates $\eta_w, \eta_P^p, \eta_P^d$. Initial Pseudolabels for unlabeled data chosen as: $y_i = e_i$ with probability $1/k$ where e_i is the one-hot vector at $i-$th coordinate.
for $O = 0, 1, 2, ..., M_O$ **do**
  Reinitialize the network parameters w^0
  $\Delta P_u = 0$
  for $I = 0, 1, 2, ..., M_I$ **do**
    (Primal) SGD Step on w: $w^{t+1} \leftarrow w^t - \eta_w \nabla L$
    (Primal) SGD Step on ΔP_u: $\Delta P_u \leftarrow \Delta P_u - \eta_P^p \nabla L$
  end for
  (Dual) SGD Step on P_u: $P_u \leftarrow P_u - \eta_P^d \Delta P_u$
end for
Output: Classification model w.

architecture to run SaaS [6] with **one crucial difference**: ReLU activations were replaced with Hermite activations. All other hyperparameters that are needed are provided in Table 1. We will call this version, **Hermite-SaaS**. To make sure that our findings are broadly applicable, we conducted experiments with four classification datasets that are commonly used in the semi-supervised learning literature: SVHN, CIFAR-10 dataset, SmallNORB, and MNIST. For the SVHN dataset, we measure pseudolabel error for both Hermites and ReLU, and compare them with the error rates reported in [6].

Dataset	# Labeled Examples	# Unlabeled Examples	Data Augmentation	# Inner Epochs / # Outer Epochs
SVHN	1,000	72,257	Affine transforms, Normalize	$\frac{5}{75}$
CIFAR-10	4,000	46,000	Affine transforms, Normalize	$\frac{10}{135}$
SmallNORB	1,000	23,300	None	$\frac{10}{135}$
MNIST	1,000	59,000	Normalize	$\frac{1}{75}$

Table 1: Dataset and implementation details used in Semi-Supervised Learning experiments.

Here, we will describe the key results, please see supplement for detailed empirical evidence.

	ReLU-SaaS	Hermite-SaaS
Error rate by SaaS on unlabeled data	6.22	**5.79**

Table 2: Error rate on unknown labels for the SVHN-1k dataset.

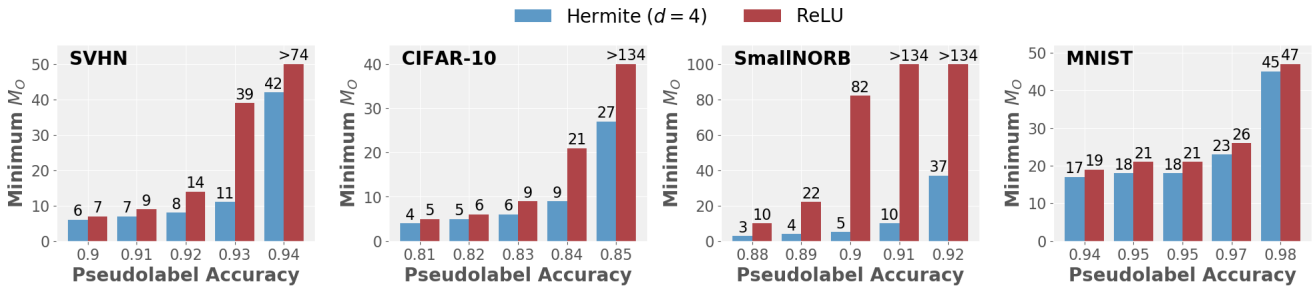


Figure 5: Hermite-SaaS trains faster. We plot the number of outer epochs M_O vs. the pseudolabel accuracy across 4 datasets. We consistently observe that the minimum number of outer epochs M_O to reach a given value of pseudolabel accuracy is always lower for Hermite-SaaS than ReLU-SaaS.

I) Computational Benefits. Table 2 shows that the quality of pseudolabels can be significantly improved using Hermite-SaaS. Moreover, we also find that the number of epochs needed to reach a specific pseudo label accuracy is also significantly lower. Table 3 shows two common metrics used in classification: (i) "pseudolabel accuracy" measures the accuracy of the pseudolabels; and (ii) "max gap in accuracy" measures the maximum difference in the pseudolabel accuracy over epochs during training. In addition, Figure 5 shows that that the number of epochs needed to reach a specific pseudolabel accuracy is **significantly lower** when using Hermite-SaaS. This behavior for Hermite-SaaS is true over all the four datsets, although the gap is quite large in the first three.

II) Smoothness boosts first order methods and provides financial savings. ReLU takes less time per iteration since in our Hermite-SaaS procedure, we must perform a few more matrix vector multiplications. However, our experimental results indicate that the lower *per iteration* time of ReLU is negligible as compared to the total number of iterations. To provide a better context for the amount of savings possible using Hermite-SaaS, we performed an experiment to assess financial savings. We AWS p3.2x large cluster for training. We calculate the cost (to train a network) by running the algorithm to reach a minimum level of pseudolabel accuracy (or a maximum number of epochs), using the default pricing model given by AWS. We can clearly see from Table 4, that we get significant cost savings if we use Hermite-SaaS. Note that we found cases where ReLU-SaaS could not reach the pseudolabel accuracy that is achieved by Hermite-SaaS: in these cases, we report a conservative lower bound for cost savings.

III) Faster Convergence of Loss functions. Recall that SaaS tries to identify labels on which the training loss decreases at a faster rate. Our experiments show that the use of Hermites provides models on which the loss function can be optimized easily, thus stabilizing the two phase procedure. Figure 6 shows the result of our experiment on CIFAR-10 dataset. Notice that the periodic jumps in the loss value is expected. This is because the big jumps correspond to the termination of an inner epoch, where the pseudolabels are updated and weights are reinitialized. Once again, from Figure 6, we observe that Hermite-SaaS provides a smoother landscape during training, accelerating the training process overall.

IV) High generalization performance of Hermite-SaaS based models. We will now analyze the performance of Hermite-SaaS based pseudolabels for the test set, that is, Phase II. Here, we use the pseudolabels generated after Phase I and perform standard supervised training using these pseudolabels. We use Resnet18 with a preactivation block for the supervised training phase, although other architectures can be utilized. Our experiments indicate that the performance of both Hermite-SaaS and ReLU-SaaS are comparable. When using Hermite-SaaS to estimate pseudolabels on the SVHN dataset, we achieve a test set error of 3.54%. This is better than the 3.82% reported by [6] which corresponds to ReLU-SaaS, but the gap is not large.

V) Compute high quality pseudolabels very effi-

		Pseudolabel Accuracies	Max Δ
SVHN-1k	Hermite	94.2%	6.1%
	ReLU	93.3%	
CIFAR-10-4k	Hermite	85.5%	3.4%
	ReLU	84.4%	
SmallNORB-1k	Hermite	92.6%	5.2%
	ReLU	90.4%	
MNIST-1k	Hermite	98.2%	4.5%
	ReLU	98.2%	

Table 3: Pseudolabel performance metrics. We also report the maximum difference Δ in Hermite and ReLU accuracies attained during training..

		Time per Epoch (sec)	Total Time (hours)	Cost ($)	Savings ($)
SVHN	Hermite	411.22	1.14	27.96	56.14
	ReLU	317.1	3.43	84.1	
CIFAR-10	Hermite	270.57	1.88	46	$\geq$ 117.2
	ReLU	243.16	$\geq$ 9.12	$\geq$ 223.2	
SmallNORB	Hermite	40.7	0.42	10.23	$\geq$ 12.97
	ReLU	25.27	$\geq$ 0.95	$\geq$ 23.2	
MNIST	Hermite	84.75	1.06	25.93	-9.13
	ReLU	51.5	0.67	16.8	

Table 4: Expenses when training Hermite-SaaS and ReLU-SaaS on AWS. We observe that although Hermite-SaaS takes more time per epoch than ReLU-SaaS, the overall gains are better for Hermite-SaaS.

ciently. From observing the performance on the test dataset, we find that a good pseudolabel accuracy is essential for higher test set accuracies. We observed that, on CIFAR10 dataset, when we run for 10 inner epochs, Hermite-SaaS results in a test set error of 12.64% relative to ReLU-SaaS which has a 13.37% error. However, Hermite-SaaS achieved a better error rate in a quarter of the training time than ReLU-SaaS.[6].

5. Noise Tolerance Using Hermite Activations

Table 5 suggest that at least based on these experiments, the confidence interval (or variance) of our Hermite activation based models tend to be small. We seek to explore this empirical behavior more carefully here and perform experiments to assess if this characteristic can be quantified as a function of *label noise*. Our experimental setting is identical to Section 4, except that here, we inject noise in the labels. In particular, we chose the label for a subset of images, uniformly at random.

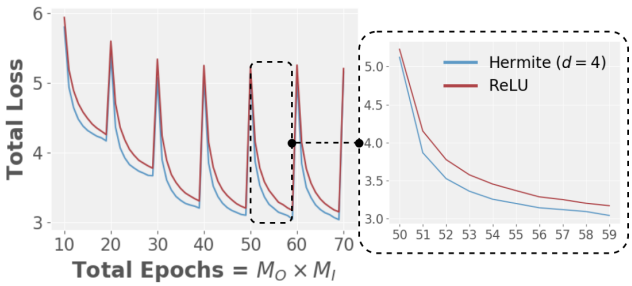


Figure 6: Convergence of loss functions in the SaaS framework. Larger spikes correspond to the end of inner loop and are due to weight reinitialization. The right plot shows that hermites accelerate training process, ensuring high quality pseudolabels.

10% Noise		A_{Best}	N_{Best}
SaaS	Hermite	85	224
	ReLU	84	328
SaaS + Tanaka *et al.*	Hermite	85	244
	ReLU	84	284

30% Noise		A_{Best}	N_{Best}
SaaS	Hermite	$\sim$80	95
	ReLU	$\sim$80	$\geq$600
SaaS + Tanaka *et al.*	Hermite	$\sim$80	61
	ReLU	$\sim$80	299

Table 5: SaaS experiments on Noisy Labelled dataset. We report the best accuracy (A_{Best}), and number of epochs (N_{Best}) to reach this accuracy. Tanaka *et al*. stands for noisy label processing method proposed in [40].

Noise injection experiments. We utilize the method proposed by [40] as an optional post-processing step after Phase I: basically, the procedure performs a "correction" to minimize the influence of noise once the pseudolabels have been estimated. The authors in [40] propose an alternating optimization algorithm where the weights of the network and the labels of the data are updated alternatively. We conduct experiments with 10% and 30% label noise levels on the CIFAR-10 dataset. After estimating the pseudolabels and/or using the scheme in [40], we trained a model in a supervised manner using Resnet18.

Our results summarized in Table 5 show that Hermite-SaaS based models obtains similar or a higher test set accuracy. This is encouraging, but we also observe that our model converges faster compared to a ReLU-SaaS model. In other words, the proposed Hermite activations based training yields models/estimators with low variance suggesting that they may behave well in the presence of outliers. Our experimental results also indicate that post processing techniques (such as [40]) may not always be usefuls to improve the generalization performance of models.

5.1. Why are Hermites noise resistant?

Based on the experimental evidence of the noise resistance of hermites, in this section, we provide theoretical justifications to support our empirical finding. Specifically, we show that when the *test data is somewhat different from the training data*, Hermite based networks do **not** produce confident predictions. This property is very important as predictions with low confidence should or can be labeled manually, for example, using a human-in-the-loop setting [44] [36]. In fact, it is well known that CNN based networks are easily fooled into making high confidence predictions [32]. Our result shows that this problem may be alleviated by using hermites, instead.

We make this argument rigorous in the remainder of this

section. In order to prove our main result, we first prove a generic pertubation bound in the following lemma. Consider a 2 layer network with an input layer, hidden layer and an output layer, each with multiple units. Denote $f_k(x)$ and $f_l(x)$ to be two output units with x denoting the input vector. Let w_j be the weight vector between input and the j^{th} and a_{lk} be the weight connecting l^{th} hidden unit to the k^{th} output unit.

Lemma 1. *Consider the output unit of the network, $f_k(x) = \sum_j a_{kj} \sum_{i=0}^{d} c_i h_i(w_j^T x)$, where $c_i's$ are the Hermite coefficients and d is the maximum degree of the herimite polynomial considered. Then,*

$$|f_l(x) - f_k(x)| \leq Cd\alpha\beta$$

Here, C is a constant that depends on the coefficients of the Hermite polynomials and

$$\alpha = \max_{lk} \sum_j |a_{lj} - a_{kj}| \; ; \beta = \max(\|w\|_p^d \|x\|_q^d, \|w\|_p \|x\|_q),$$

such that $1/p + 1/q = 1$.

Proof. Sketch. The proof proceeds by using elementary techniques such as Cauchy-Schwarz and Holder's inequality. Only the asymptotic dependence on number of polynomials d is important. □

Now, we use the perturbation bound from Lemma 1 to show the following result (proof in the supplement) that characterizes the behavior of a network if the test example is "far" from examples seen during training.

Theorem 2. *Let $f_k(x) = \sum_j a_{kj} \sum_{i=0}^{\infty} c_i h_i(w_j^T x)$, be a one-hidden layer network with the sum of infinite series of hermite polynomials as an activation function. Here, $k = 1, 2, ..., K$ are the different classes. Define $w_J = \min w_j^T x$. Let the data x be mean normalized. If $\epsilon > 0$, the Hermite coefficients $c_i = (-1)^i$ and*

$$\|x\| \geq \frac{1}{\|w_J\|} \log \frac{\alpha}{\log(1 + K\epsilon)}$$

then, we have that the predictions are approximately (uniformly) random. That is,

$$\frac{1}{K} - \epsilon \leq \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}} \leq \frac{1}{K} + \epsilon \; \forall \; k \in \{1, 2..., K\}$$

Proof. Sketch. Note that the form of coefficients is important for this theorem. In particular, we use the exponential generating functions expansion of hermites and exploit the form of normalization due to softmax layer to provide a lower bound for the confidence of prediction for an arbitrary class. For this event to occur with high probability, we show that the test data has to be at least a certain distance far away from training data. □

As the data is mean normalized, any test example x with high $\|x\|$ implies that it is far from the training data. For large $\|x_{\text{test}}\|$, the above theorem shows that the predictions are fairly random, which is a desirable property – clearly, we should not provide a confident prediction when we have not seen such an example earlier. We point to an interesting trade-off between computational complexity that we observe from our experiments and statistical complexity from theory: Theorem 2 requires d to be very large whereas training becomes intractable in those regimes.

Our lemma 1 (proof in the supplement) is similar to the perturbation bounds that appear in learning theoretic results from a few years back, see for example, Lemma 2 in [31]. Hence, it may be possible to extend these generalization results to our hermite activation based networks as well.

6. Conclusion

In this paper, we studied the viability and potential benefits of using a finite Hermite polynomial bases as activation functions, as a substitute for Rectified Linear Units (ReLUs). The lower order Hermite polynomials are known to have nice mathematical properties from the optimization landscape point of view, although little is known in terms of their practical applicability to networks with more than a few layers or to interesting tasks in vision. We observed from our extensive set of experiments that simply replacing ReLU with an expansion in terms of Hermite polynomials can yield significant computational benefits, and we demonstrate the utility of this idea in a computationally intensive semi-supervised learning task. Under the assumption that the training is being performed on the cloud and with published pricing structure, we show sizable financial savings are possible. On the mathematical side, we also showed that Hermite based networks have nice noise stability properties that appears to be an interesting topic to investigate, from the robustness or adversarial angles.

References

[1] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

[2] S. Arora, N. Cohen, and E. Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *ICML*, 2018.

[3] J. Bergstra, G. Desjardins, P. Lamblin, and Y. Bengio. Quadratic polynomials learn better image features. *Technical report, 1337*, 2009. 3

[4] J. P. Boyd. Asymptotic coefficients of hermite function series. *Journal of Computational Physics*, 54(3):382–410, 1984. 2

[5] F. Chollet et al. Keras. https://keras.io, 2015. 3

[6] S. Cicek, A. Fawzi, and S. Soatto. Saas: Speed as a supervisor for semi-supervised learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 4, 5, 6, 7

[7] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 1

[8] G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013. 1

[9] X. Dong, G. Kang, K. Zhan, and Y. Yang. Eraserelu: a simple way to ease the training of deep convolution neural networks. *arXiv preprint arXiv:1709.07634*, 2017.

[10] S. S. Du and J. D. Lee. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

[11] R. Ge, J. D. Lee, and T. Ma. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*, 2017. 2

[12] R. Ge, J. D. Lee, and T. Ma. Learning one-hidden-layer neural networks with landscape design. In *International Conference on Learning Representations*, 2018. 3

[13] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. 1, 3

[14] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013. 2

[15] S. Guarnieri, F. Piazza, and A. Uncini. Multilayer feedforward networks with adaptive spline activation function. *IEEE Transactions on Neural Networks*, 10(3):672–683, 1999.

[16] M. Hardt and T. Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[19] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 3, 5

[20] M. Hein, M. Andriushchenko, and J. Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *arXiv preprint arXiv:1812.05720*, 2018.

[21] J. Jang, J. Kim, J. Lee, and S. Yang. Neural networks with activation networks, 2018.

[22] L. Kargin and V. Kurt. On generalized humbert matrix polynomials. *Miskolc Mathematical Notes*, 2014. 2

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[24] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015. 1

[25] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013. 4

[26] L. Ma and K. Khorasani. Constructive feedforward neural networks using hermite polynomial activation functions. *IEEE Transactions on Neural Networks*, 16(4):821–833, 2005.

[27] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013. 1

[28] L. B. Michael, M. Ghavami, and R. Kohno. Multiple pulse generator for ultra-wideband communication using hermite polynomial based orthogonal pulses. In *2002 IEEE Conference on Ultra Wideband Systems and Technologies (IEEE Cat. No. 02EX580)*, pages 47–51. IEEE, 2002. 2

[29] E. Mossel, R. O'Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, 2005. 2

[30] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[31] B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017. 8

[32] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015. 7

[33] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3239–3250, 2018. 4

[34] F. Piazza, A. Uncini, and M. Zenobi. Artificial neural networks with adaptive polynomial activation function. 1992.

[35] A. Rasiah, R. Togneri, and Y. Attikiouzel. Modelling 1-d signals using hermite basis functions. *IEE Proceedings-Vision, Image and Signal Processing*, 144(6):345–354, 1997. 2

[36] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016. 7

[37] W. Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 2006. 2

[38] A. Shah, E. Kadam, H. Shah, S. Shinde, and S. Shingade. Deep residual networks with exponential linear unit. In *Pro-*

ceedings of the Third International Symposium on Computer Vision and the Internet, pages 59–65. ACM, 2016.

[39] S. M. Siniscalchi, J. Li, and C.-H. Lee. Hermitian based hidden activation functions for adaptation of hybrid hmm/ann models. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[40] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018. 7

[41] Y. Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3404–3413. JMLR. org, 2017. 2

[42] C. Wei, J. D. Lee, Q. Liu, and T. Ma. On the margin theory of feedforward neural networks. *arXiv preprint arXiv:1810.05369*, 2018. 2

[43] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 1

[44] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 7

[45] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 4